(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号

特開平5－250515

(43)公開日　平成５年(1993)９月28日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｋ　9/20	３６０　Ａ			
7/10	Ｒ	8945－5Ｌ		

審査請求　未請求　請求項の数２(全 14 頁)

(21)出願番号　特願平4－267871

(22)出願日　平成４年(1992)９月11日

(31)優先権主張番号　７６１６６１

(32)優先日　1991年９月18日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　592089054
エヌ・シー・アール・インターナショナル・インコーポレイテッド
アメリカ合衆国　45479　オハイオ、デイトン　サウス　パターソン　ブールバード　1700

(72)発明者　デニス　ダブリュー．ゴーレム
カナダ　エヌ２ケイ　３ビー５、オンタリオ、ウォータールー、ダンズバリ　ドライブ　368

(74)代理人　弁理士　西山　善章

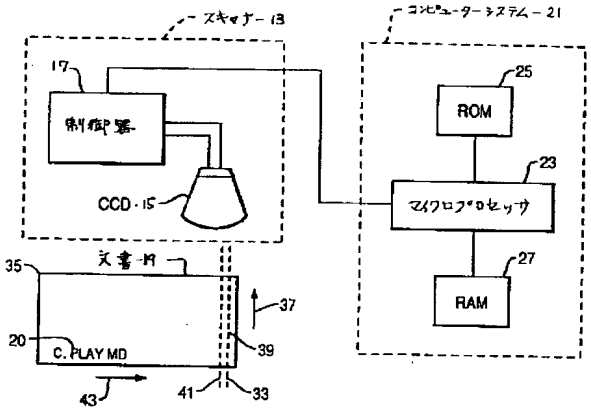
最終頁に続く

(54)【発明の名称】　テンプレートを使用してバーコード文字を光学的に認識する方法および装置

(57)【要約】　（修正有）

【目的】　小切手等の文書上のバーコード文字の光学的走査と認識を行う方法とシステムを提供する。

【構成】　本システム２１は光学的スキャナ１３、マイクロプロセッサ２３、ＲＯＭ２５とデータ並びに複数の予定文字同定２進パターン（テンプレート）を格納する格納プログラム付ＲＡＭ２７を含む。スキャナは文書１９を走査し、複数のグレイスケールピクセル値を発生。マイクロプロセッサ２３は格納されたピクセル値を処理し各文字２０の位置を特定し、その文字をテンプレートに比較照合。プロセッサは照合の失敗、文字の拒絶時に、文字の座標を使用して当該文字をセグメント化した文字のピクセル値を２進データに変換し、これを圧縮し、テンプレートと照合し、当該２進データに符合するテンプレートを複数テンプレートから選択する。



1

【特許請求の範囲】

【請求項１】光学的文字認識システムであって文書上の不透明度の異なるバーおよび間隙からなる予定パターンを有するバーコード文字を光学的に走査すると共に、該不透明度に対応するグレイスケール値を、明ピクセルおよび暗ピクセルのパターンを表す複数のグレイスケール値として、発生する手段と、

該文字を表す予定の２進パターンを含む複数の予定２進パターンを含む予定テンプレートと該グレイスケール値とを格納するためのプログラムを具えたメモリ手段と、該走査手段および該メモリ手段に結合され、該プログラムの制御の下にオペレーションを行う処理手段にして、該グレイスケール値から該文字を表す値のマトリックスを選択することにより該文書上の該文字の位置を特定し、該マトリックス値を２進データに変換し、該２進データを圧縮し、さらに該圧縮済み該２進データに符合することにより該文字を同定する働きをするパターンを該複数の予定テンプレート２進パターンの中から選択する、前記処理手段とを含む光学文字認識システム。

【請求項２】予定長の高さ、幅および不透明度の異なる予定のバーと間隙を有する文書上のバーコード文字にして複数の予定テンプレート２進パターンの一つにより表される前記バーコード文字を、光学的に同定する方法にして、

光学的に文書を走査して該不透明度に対応するグレイスケール値を、明ピクセルおよび暗ピクセルを表す複数のグレイスケール値として、発生するステップと、

該グレイスケール値から文字を表す値のマトリックスを選択することにより該文書上の文字の位置を特定するステップと、

該マトリックス値を２進データに変換するステップと、

該２進データを圧縮するステップと、

該圧縮済み該２進データに符合することにより該文字を同定する働きをするパターンを該複数の予定テンプレート２進パターンの中から選択するステップとを含むバーコード文字同定方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は文書の走査および英数字一般の認識に関し、特にテンプレートを使用してＣＭＣ７型のバーコード化された文字（以下、バーコード文字という）を光学的に走査し認識することに関する。

【０００２】

【従来の技術】ＣＭＣ７文字セットはフランスのカンパニエ・ド・マシネ・ブル社により開発された文字組みで、主としてヨーロッパの多くの国で経済取引上広く使用されている。一般的にこれらの文字は銀行為替、小切手その他の経理文書上に磁気スキャナで走査できるＭＩＣＲ（磁気インク文字記録）形式で記録されている。各文字は、予定の高さ、幅、および不透明度の異なるバー

2

とバー間の間隙（例えば７本のバーとバー間の６個の間隙であって、４個の間隙が短い（狭い）幅を有し、２個の間隙が長い（広い）幅）を有する。しかしながら、バーコードの印刷品質が劣っていたり、当該文字パターンがぺんによるなぞり書きその他のノイズにより崩れているときは光学的文字認識はしばしば適切な文字認識ができない。

【０００３】

【発明が解決しようとする課題】そこで本発明は、文字パターンの印刷が低品質のとき、あるいは文字パターンのバー間隔に不規則その他の文字崩れの問題があるとき、それらバーコード文字が良好に認識できるシステムを与えることを課題とする。

【０００４】

【課題を解決するための手段】本発明は、文字形状（テンプレートパターン）情報を使用してバーコード文字を認識する光学的文字認識のためのシステムおよび方法を与える。このシステムは、印刷が低品質であるとき、あるいは文字の一部が不明瞭であったり、その他外部的原因で崩れている場合でも、良好にバーコード文字を認識できる。

【０００５】第一の局面で、本発明の光学文字認識システムは、（ａ）文字と異なる不透明度を有する文書上において不透明度の異なるバーおよび間隙からなる予定パターンを有するバーコード文字を光学的に走査（入力および格納）すると共に、該不透明度に対応するグレイスケールピクセル値を、明ピクセルと暗ピクセルを表す複数のグレイスケールピクセル値として発生するＣＣＤスキャナと、（ｂ）前記グレイスケールピクセル値と、プログラムと、文字を表す複数の予定テンプレート２進パターンにしてその一つが該文字を表すようにされたテンプレート２進パターンとを格納するメモリと、（ｃ）前記スキャナおよびメモリに結合され前記プログラムの制御の下にオペレーションを行うプロセッサにして、該グレイスケール値から該文字を表す値のマトリックスを選択することにより該文書上の該文字の位置を特定し、該マトリックス値を２進データに変換し、該２進データを圧縮し、さらに該圧縮済み該２進データに符合することにより該文字を同定する働きをするパターンを該複数の予定テンプレート２進パターンの中から選択するプロセッサを含む。

【０００６】このＣＣＤスキャナは制御論理回路を含み、プロセッサはマイクロプロセッサを含み、メモリは立ち上げルーチン付きＲＯＭ（読み取り専用メモリ）と、プログラム、ビクセル値および２進パターンを有するＲＡＭ（ランダムアクセスメモリ）とを含む。

【０００７】第二の局面で、本発明の方法は、予定長の高さ、幅、および不透明度の異なる予定のバーと間隙を有する文書上のバーコード文字にして複数の予定テンプレート２進パターンの一つにより表される前記バーコー

ド文字を光学的に同定する方法にして、（ａ）光学的に文書を走査し、該不透明度に対応するグレイスケールピクセル値を、明ピクセルおよび暗ピクセルを表す複数のグレイスケールピクセル値として発生するステップと、（ｂ）該グレイスケールピクセル値から文字を表す値のマトリックスを選択することにより該文書上の文字の位置を特定するステップと、（ｃ）該マトリックス値を２進データに変換するステップと、（ｄ）該２進データを圧縮するステップと、（ｅ）該圧縮済み該２進データに符合することにより文字を同定する働きをするパターンを該複数の予定テンプレート２進パターンの中から選択するステップとを含むバーコード文字同定方法を与える。文書上の文字は例えばＣＭＣ７フォーマットで記録されている。カメラ（ＣＣＤスキャナ）は文書を鉛直方向に、底部右隅から出発して底部から頂部へ走査する。文書は文書の左頂部隅に走査が到達し走査が終了したことを示すまで、文書は右へ連続的に移動される（その場合、鉛直方向の各走査幅は一ピクセルの直径に相当する）。

【０００８】ある文字（すなわちある文字を表すグレースケールピクセル値）の位置を特定するにあたり、本システムは当該文字に対して選択したピクセル値の上に窓（ウインドウ）を配置する（以下これを、文字のフレーム化と言う）。このフレーム化はこのウィンドウ内の選択ピクセル値の和が最大となるように行う。ウインドウの配置に当たり、本システムは文字データの少なくとも三列を使用して文字縁および文字間間隙を同定する。次に本システムは選択したピクセル値から、しきいレベルを計算し、選択したピクセル値をそのしきいレベルを基準とする２進データに変換する。次にこの２進データはより小さなマトリックス（配列）に圧縮され、その文字と最も良く一致するパターンが発見される迄、予定のテンプレート２進パターンの各々と論理比較される。各テンプレート（２進パターン）は三ビットコンフィギュレーション（レベル）を有する。その第一ビットコンフィギュレーションは文字の実際のビットパターンを表し、第二ビットコンフィギュレーションは実際のビットパターンのビットの有意性（significance）を表し、第三ビットコンフィギュレーションは実際のビットパターンのビットの重み値を表す。この変換された２進データが各予定ビットパターンの各三ビットコンフィギュレーションに論理的に匹敵する程度が、「ミスマッチワード」として表現され、このミスマッチワードからミスマッチ計数が計算される。次にこのミスマッチ計数が少なくとも一つの予定しきい値と比較され、当該しきい値より小さい（すなわちしきい値未満の）最小ミスマッチ計数が当該文字を表すものとして選択される。

【０００９】

【実施例】以下に本発明のシステムを説明する。本システムは印刷が低品質であるかあるいは文字の一部（バー、間隙）が外部的原因で崩れているときに良好なバーコード文字認識を与える。

【００１０】文字間のバー間隔に関する情報は従来バーコード文字を認識するのに使用されている。文字形状およびテンプレートパターン情報は従来バーコード化されていない文字の認識に使用されている。本発明のシステムはテンプレートパターン情報をバーコード文字の認識に拡張使用するものである（以下、この方法をテンプレートマッチング方法という）。テンプレートマッチング方法の一適用例（ルーチン）を挙げれば、バーコード認識方法を使用して行うＣＭＣ７バーコード文書の読み取りがある。この方法は、印刷品質が低いため、または外部的ノイズのためにバーコード文字の認識に失敗したとき、その文字の分類を助けるため、本テンプレートマッチング方法が呼び出される。テンプレートマッチング方法が当該文字の分類を完了した後、バーコード認識方法が続行され、文書の残りが処理される。その後、必要に応じてテンプレートマッチング方法が呼び出される。

【００１１】図１は本発明のシステム１１を示す。このシステムは文書１９を走査するための光学的読取り機またはスキャナ１３と、スキャナ１３から受信したデータを処理するコンピューターシステム２１とを含む。このスキャナはＣＣＤ（電荷結合デバイス）１５および制御器１７を具えたカメラでよい。文書１９は銀行為替とか上に英数字２０を印刷された小切手等である。コンピューターシステム２１はマイクロプロセッサ２３と、ブートプログラムを備えたＲＯＭ（読み取り専用メモリ）２５と、走査デバイスおよびマイクロプロセッサからのデータを格納し、また予め定義されたテンプレートパターン（すなわち選択された文字を表す２進データのパターン）を格納するためのプログラム付きＲＡＭ（随時アクセス可能メモリ）２７を含む。

【００１２】走査オペレーションは文書１９の右底部隅３３から始まり、頂部左隅３５で終わる。スキャナ１３は文書１９の底部から頂部に向けて鉛直方向に、最初は図の矢印３７で示すように最右の走査線３９から走査し、次いで再び底部から頂部へ次の最右の走査線４１に沿って走査を行い、以下同様にして文書の頂部左隅に達するまで走査する。走査線は平行に配列されている。文書１９は矢印４３で示すように左から右へ連続的に移動される。スキャナは連続的走査線に沿って走査を行う。各走査線の幅はピクセルの直径（約０．０１２７ｃｍすなわち０．００５インチ）に相当する。このようにして連続する文字が文書から読み取られる（すなわち入力され、格納される）。

【００１３】図２に示すような幅２１ピクセル、高さ２８ピクセルの文字の場合、本システムは、文字をあたかもそれが幅２３ピクセル、高さ３０ピクセルの領域内にあるかのように、すなわち文字の頂部、底部および両側に１ピクセルの白縁があるかのように、取り扱う。文書

上に記録された文字のイメージは磁気的なものでも非磁気的なものであってもよい。

【００１４】図２に示すように各文字２０は７個の平行な鉛直バー（データバー）４５と４個の短間隙４７とバー間の二つの長間隙４９とで一意的に定義される。走査方向３７（図１）は平行なバーの方向に一致する。各バーは０．０１０ないし０．０１９ｃｍの幅で、各短間隙は０．０１１ないし０．０１９ｃｍの幅、各長間隙は０．０３１ないし０．０３９ｃｍの幅でよい。

【００１５】文字の走査および処理（セグメント化および認識）は、各文字を構成する鉛直バーと間隙を含む当該文字形状とで記述される。各バーの高さは変化するが、バーおよび間隙の幅は予定の範囲内に保持（印刷）される。一文字についてのバーの幅および間隙は同じ文字組の他の文字のバーおよび間隙の幅と同一である。

【００１６】文字のいろいろのバーおよび間隙を表すため、文字が走査されるときにスキャナにより発生されるピクセル値は、例えば２８行（すなわち文字の高さを表す２８ピクセル）および２１列（すなわち文字幅に相当する２１ピクセル）を有するデータ配列（マトリックス）としてメモリ２７（図１）中に格納される。連続する文字間の空間および文字の上方と下方の空間（すなわち文書の地が現われている空間）を表すピクセルを除外すると、文字（例えば文字「９」）のバーおよび間隙は図３に示す配列で表すことができる。各暗ピクセル５１は鉛直バー４５の一部を表し、範囲０-２５５の不透明インデックスすなわちグレイスケール値を有する。各明ピクセル５３は間隙４７、４９または文字間空間５５の部分を表し、範囲０-２５５の不透明／グレイスケール値を有する。暗ピクセルは一般に大きなグレイスケール値（２５５に近い値）を有し、また明ピクセルは一般により小さな値（０に近い値）を有する。

【００１７】上述したようにこれらのグレイスケールデータ値は文書を走査するスキャナにより発生される。発生されたデータは次いでシステムで処理するため、メモリ２７（図１）内に格納される。バーコード文字データの処理において本システムで採用する方法は、「バーコード文字の処理」という以下の見出しつけた下記の項で説明する。しかし読者にとって本処理方法をより良く理解できるようにするため、バーコード化されていない文字データ（以下、非バーコード文字データという）の処理を初めに説明する。

【００１８】非バーコード文字データの処理

このシステムは格納されたプログラムの制御の下に、格納されたデータ値を調べ、最初の文字の位置特定に進み、次いで最初のフィールドの後続文字に進み、次いで後続フィールドの他の文字に進む。文字の位置特定が終わると、システムは当該文字をフレーム化または区画化（delimit）する（すなわち他の文字から当該文字を分離またはセグメント化（seqment）する）。最右のスキャナ走査線３９（図１、２）に沿って底部から頂部への走査から開始し、次いで次の最右走査線４１に沿って底部から頂部へ走査し、以下同様に文書を走査して得た格納グレイスケールデータ値から、システムは以下に述べるように各文字の位置を特定しフレーム化（セグメント化）する。これは、最初に格納されたデータを捜索し、走査線に沿う明ピクセルから暗ピクセルへの遷移を表すグレイスケール値の上昇を探知することにより行う。もしも例えば４０（この４０というこの特定の値はフォントに依存する）より大きい上昇が、走査方向３７に隣接するピクセルの二つのグレイスケールデータ値の間に発見されると、暗ピクセルが発見されたと考えられる。この暗ピクセルに近接する（文書の）領域は、文字が発見される確率が高い領域であると見做される。暗ピクセルが発見されると、システムはこの文字領域を検査し（すなわち暗ピクセル値に近い格納済みデータを検査し）、この暗ピクセルが文字の一部であるか否かを決定する。システムは文書上の文字を効果的にフレーム化するに足りる大きさのセグメント化ウィンドウを確立することにより文字の存在を検査する。暗ピクセルの位置はウィンドウの中心に相当する。左縁および右縁（列）５５、５７および頂部縁および底部縁（行）５９、６１を有するセグメント化ウィンドウ５３が図４（ａ）に示されている。左縁および右縁の境界をなしているのは（左＋１）列６３および（右－１）列６５で、頂部縁および底部縁の境界をなしているのは（最上＋１）行６７、および（底部－１）行６９である。

【００１９】図４（ａ）、５を参照して以下に説明するように、文字はウィンドウ内のすべてのグレイスケールピクセル値の和が最大となったときにフレーム化され、かつウィンドウの中心に配置されたと考えられる。ウィンドウが移動されるときに和（Ｓｗ）が増大するか否かを決定するため、左右の列５５、５７各々の中のピクセル値とそれらの各境界列６３、６５の中のピクセル値との和が計算される。同様に、頂部列５９および底部列６１各々の中のピクセル値と境界列６７、６９各々の中のピクセル値との和が計算される。例えばもしも列６３内の和が列５７内の和よりも大きいことが発見され、かつ列６３を含め列５７を排除するようにウィンドウ５３を一ピクセルだけ左へ移動するとウィンドウ内の和（Ｓｗ）が増大するようであれば、ウィンドウは左へ移動される。同様に、もしも行６７を含め、行６１を排除するようにウィンドウを一ピクセルだけ上に移動すればウィンドウ内の和（Ｓｗ）が増大するなら、ウィンドウは上に移動される。このようにウィンドウに隣接する（すなわち境界をなす）行、列のピクセル値の和（累積的カウント）は、ウィンドウの縁にある行および列のピクセル値の和に相対的な値であり、この値からウィンドウ内のピクセル値の和（Ｓｗ）を増大させるようにウィンドウを移動させる方向が決定できる。図５に示すように、下

記のカウントにより、左への移動に伴ってウィンドウカウントが１００だけ増大され、上への移動に伴って１７０増大される。

「右－１」列のカウント　＝　０
「右」列のカウント　＝　０
「左」列のカウント　＝　１００
「左＋１」列のカウント　＝　１００
「底部－１」行のカウント　＝　０
「底部」行のカウント　＝　０
「頂部」行のカウント　＝　１７０
「頂部＋１」行のカウント　＝　１７０

したがってウィンドウは左および上へ移動される。このようにしてウィンドウはＳｗの増大する方向に移動され、このオペレーションはウィンドウを四つの任意の方向（左、右、上、下）に移動してももはやＳｗを増大させなくなるまで反復される。このときウィンドウは文字をフレーム化した（区画化またはセグメント化した）と考えられ、当該文字はウィンドウの中心に配置されていると考えられる。このようにしてこのウィンドウにより文字を最もよく表すピクセル値のマトリックスが選択される（フレーム化される）。

【００２０】フレーム化された文字を表す情報（暗ピクセルグレイスケール値）が十分にウィンドウ内にあるか否かを決定する検査として、予定の確認しきい値（例えばしきいグレイスケール値１００）を超えるウィンドウ内ピクセルの全数を計算する。もしもこの全数が例えば６０（Ｅ１３Ｂフォントのようなフォントの場合）以上であると、ウィンドウ内に文字が見つかったことが確認される。そうでないときはウィンドウ内にフレーム化された対象物はノイズと解釈され、システムは新しい文字を発見する手続きに進む。

【００２１】フレーム化の後、セグメント化（フレーム＊化）された文字は同定に備えて縮尺変更（scale）される。認識プロセスでは１６ビットワードが使用される（すなわち、テンプレートの幅が１６ピクセルである）。多くの文字が１６ピクセルより幅が広い（たとえばＥ１３Ｂ文字は１９ピクセルである）ので、それらはフレーム化した後に１６ビットに変換される。これは予定の列、例えばＥ１３Ｂフォントの場合、列４、９、および１４を除去することにより達成される。（ＯＣＲＡおよびＯＣＲＢフォントは、２００ピクセル／インチのとき、幅が１６ピクセルであり、列の除去は必要でない。）

【００２２】前節までの説明は処理の準備として文字がどのように文書上の位置特定を受けるか、すなわち後の処理のためどのように文書上の位置を特定され効率よくフレーム化（セグメント化）されるかを説明するものである。以下の説明は、フレーム化された文字が当該文字の同定（分類／認識）のために２進型式に変換され、次いでテンプレート（これは予定の２進パターンである）と比較される一つの方法の説明である。

【００２３】この変換オペレーションではフレーム化された各行の文字について格納されたグレイスケールピクセル値が２進型式に変換される。最初に、フレーム化された文字の暗ピクセルの算術平均として基準しきい値が（前述したように）計算される。次いでこの基準しきい値に相対的に、グレイスケールピクセル値が２進形に変換される。例えばもしも計算された基準しきい値が１００であると、グレイスケール値８０、１２０、１３０、９０、８５、７０、１１０、１３５は図６の８ビットワード７１に示すビットパターン０１１００１１を有する２進データに変換することができる。これは以下に示す２進等価スケールを用いて行われる。

グレイスケールピクセル値	２進型式等価値
≧１００	１（黒）
＜１００	０（白）

【００２４】この２進データはフレーム化された文字の一行または薄片（slice）を表す。次にこの２進データおよび当該文字を表す他の行の変換済み２進データが複数のテンプレートと比較される。各テンプレートは一文字に対応する。各テンプレートは２進データの予定パターンを表し、三つの異なるレイヤー（ビットコンフィギュレーション）を含む。図６に示すこれら三つの８ビットワード７３、７５、７７は三つの各レイヤーにおける８ビット行を表す。第一レイヤー（これはビットパターンレイヤーで、パターンワード７３に対応する）は文字の実際の黒／白（０／１）パターンを表す。第二レイヤー（これはビット有意レイヤー（bit-significance layer）で、ビット有意ワード７５に相当する）は有意の文字のビット位置および有意でない文字の位置を同定する。１-ビットは、文字のビットパターンが文字の大きさおよび形状に拘わらず不変に留まると予想される有意ビット位置を同定する。０-ビットは、文字の大きさまたは形状が異なるときビットパターンが同一とは限らない非有意ビット位置を同定する。第三レイヤー（これはビット重みレイヤーで、主にワード７７に対応する）は類似の（実質的に類似の）文字を識別するときにいずれのビットが重要であるか、またそれゆえにいずれのビットが他のビットより大きな重みを有するかを同定する。この第三レイヤーでは１-ビットは重みのあるビットに指定され、０-ビットは重みなしのビットに指定される。例えば文字「Ｏ」と「Ｕ」に対するテンプレートの第三レイヤーの頂部行における１-ビットは、これら二つの文字を識別するのに有用である。また文字「Ｑ」と「Ｏ」に対するテンプレートの第三レイヤーの底部行内の１-ビットは、「Ｑ」と「Ｏ」とを識別するのに有用

である。

【００２５】図６に示すように文字同定のため、２進データ７１にいろいろの論理オペレーションがマイクロプロセッサ２３（図１）により行われる。（これら論理オペレーションは実際には１６ビットオペレーションとして行われる。しかし、簡単のため８ビットオペレーションを図示する）。第一に、２進データはビットパターンレイヤーの対応ワード７３との間で排他的ＯＲ演算され、２進データ７１と正しい予定パターン７３との間のミスマッチパターン７４を発生する。次いでこのミスマッチパターン７４はビット有意レイヤーのワード７５と論理的ＡＮＤ演算され、ミスマッチワード７６が発生される。ミスマッチワード７６は、文字の大きさおよび形状に独立なミスマッチ（不一致）の程度を表す。このミスマッチワード７６は次いで重みを付けられる。すなわちビット重みレイヤー内の対応するワード７７と論理的ＡＮＤ演算され、重み付きミスマッチインジケーター７８を発生させる。（もしも重み付きワード７７が全く１-ビットを含まないならば、（その結果は０なので処理時間を節約するため）重み付けまたはＡＮＤ演算はなされないで、システムは２進データの次行の処理に進む）。

【００２６】重み付けオペレーションに続いて、２進データ行７１に対してミスマッチカウント（MC_R）が計算される。この計算はマイクロプロセッサ２３（図１）により次式

$$MC_R = MC_W + (WOC \times WF) \quad (1)$$

で求められる。ここにMC_W　はミスマッチワードに対するミスマッチカウント（すなわちミスマッチワード７６内の１の数）、ＷＯＣ　はミスマッチインジケーターすなわち重み出力カウント（すなわちミスマッチインジケーターワード７８内の１の数）、ＷＦ　はＥ１３Ｂ型文字に対する予定の重み因子（例えば整数２）を表す。このようにして、図６のミスマッチワード７６、７８に示すミスマッチビットパターンの場合、MC_W＝２、ＷＯＣ＝１、ＷＦ＝２であることが了解できよう。それゆえ、２進データ行７１について計算したミスマッチカウント（MC_R）は上式（１）から４（すなわち２＋（１×２））となる。

【００２７】一行についてミスマッチカウントを計算した後、システムは上記の方法で文字の残りの行すべてについてのミスマッチカウントの計算に進む。次に文字のすべての行に対するミスマッチカウントが加算されてテンプレートミスマッチカウント（すなわちフレーム化された文字に適用したテンプレートに対するミスマッチカウント）を発生する。同様の方法で処理対象のフォントの文字テンプレートについても、フレーム化した文字に相対的なミスマッチカウントがテンプレート毎に発生される。いろいろのテンプレートが処理されるに従い、二つの最低テンプレートミスマッチカウントとそれらに関連したテンプレート番号とがメモリ内に格納される。文字を同定する前提条件は：もしもこの最低カウントがしきい値（Ｅ１３Ｂフォントでは拒絶しきい値４０）未満であり、次に低いカウントよりも予定量（Ｅ１３Ｂフォントではしきい値差５）だけ小さければ、システムはこの文字を同定する。

【００２８】前述したテンプレートの認識（以下、テンプレート認識という）は図７に示すように全部で９箇所で行うことができる。これはイメージ内のノイズ等の因子により完全にはフレーム化されない文字を同定するために行われる。図７に示す例では、テンプレートおよび（２進データの）入力パターンの大きさは２３行（高さ）×１６列（幅）である。テンプレートの三つのレイヤー（パターンレイヤー、ビット有意レイヤーおよび重み付けレイヤー）はすべて同一方向に同一量移動（shift）される。後述する移動位置の説明において「テンプレート行」とはビットパターン行、ビット有意行、およびビット重み付け行を指す。

【００２９】中央位置８７ではテンプレート９６は入力パターン９８の真上に配置されている。　行２ないし２２のみがこの比較に使用される。従って、テンプレート行１はパターン行１と比較され、テンプレート行２はパターン行２と比較され、・・・以下同様である。中央水平位置８１、８７、９３ではすべての列（１ないし１６）が使用される。上方中央位置８１では、テンプレートの行１ないし２２が入力パターンの行２ないし２３と比較される。これはテンプレートパターンを鉛直方向上に一行だけ移動することと同一である。この場合、テンプレート行１は入力パターン行２と比較され、テンプレート行２は入力パターン行３と比較される、等々である。この状況（水平方向の中央位置）ではテンプレートと入力パターンのすべての列が比較される。同様にして、下端中央位置９３ではテンプレートは下方に一行だけ移動されており、従ってテンプレート行２ないし２３が入力パターン行１ないし２２と比較照合される。

【００３０】水平移動した諸位置は入力パターン９８上でテンプレート９６を左または右へ移動することに対応する。中央左位置８５ではテンプレート列１ないし１５および入力パターン列２ないし１６が使用される。（これは鉛直方向に中央位置なので行２ないし２２が入力パターンおよびテンプレートの両方に使用される。）従ってテンプレートワードビット１はパターンワードビット２と比較され、テンプレートワードビット３はパターンワードビット４と比較され、以下同様に比較される。例えばもしもテンプレートパターンレイヤーワードが

００１１１１１００００１１１１

であり、入力文字パターンワードが

００１０１１１１００００１１１０

であると、テンプレートが左へ１ビット（列）移動され

る。
００１１１１１００００１１１１
排他的ＯＲ演算した結果は
０１０１０００１００００００００
となろう。左への移動演算が行われると、最右のビット（最下位ビット、ＬＳＢ）は０であることに注意されたい。従ってビット有意ワードのＬＳＢもまた０である（なぜならば三つのレイヤーすべてが同一方向に同一量移動されるからである）。それゆえ、ミスマッチワード７６のＬＳＢは（ビット有意ワードとのＡＮＤ演算の後）常に０となる。同様に、右への移動は最左ビット（最高位ビット、ＭＳＢ）を０にし、従ってミスマッチワード７６のＭＳＢは（ＡＮＤ演算の後）常に０となる。

【００３１】右向き水平移動（位置８９）は左向き移動と同様で、方向が反対なだけである。従ってテンプレートワードのビット２は入力パターンワードのビット１と整列する（比較される）。

【００３２】隅位置（位置９５、９１、８３、７９）は一行分の鉛直移動と一列分の水平移動との組み合わせを表す。例として、左上位置７９ではテンプレート行１は左へ一ビット位置だけ移動され、パターンワード２と比較され、テンプレート行２は左へ一ビット移動されてパターンワード３と比較される、等々である。

【００３３】この方法を使用して文字を認識するためには、一テンプレートあたり９回のパス数（推移数）に文字セット中のテンプレート数を乗じた数のパス数が必要である。例えば４９文字を持つ英数字セットは全部で４４１（９×４９）個の比較照合を必要とする。これは時間を消耗する。文字同定もっと速く促進するには入力パターンに対して当初は各テンプレートの中央位置のみを比較する。いろいろのテンプレートを処理するにともない、最低の二つのミスマッチカウントおよびそれらに対応するテンプレート番号をＲＡＭ２７内に格納する。この最初のパスが終ってから、ただ二つのテンプレートの残りの８個の位置を処理し、入力されたパターンに対して最良の（すなわち最低の）全ミスマッチを発見する。従って通常、たった６５個（１×４９＋２×８）の比較が必要である（すなわち必要な４４１個の比較のうちの約１５％である）。

【００３４】文字の受理または拒否を行うための前提条件を以下に述べる。

【００３５】前記最低テンプレートミスマッチカウントは、当該テンプレートが表す文字に対する予定の拒否しきい値と比較される。もしもこの文字についてのテンプレートミスマッチカウントがこのしきい値未満であり、かつ二つの最低テンプレートについてのテンプレートミスマッチの差（最低ミスマッチを、次に最低のミスマッチから減じた値）が予定の量（例えば１０）より大きいと、最低のテンプレートミスマッチを持つテンプレートがフレーム化された文字を同定する。ミスマッチカウントが近すぎるか否かを検査する理由は非常に似かよった文字（例えばＱとＯ）を区別するためである。もしもミスマッチが近いと、文字は誤った同定を犯す危険を避るために拒否される。もしも上記の拒否しきい値未満のテンプレートミスマッチがないか、または二つのテンプレートミスマッチカウント近すぎると、下に述べる後処理オペレーション（post precessing operation）が行われる。もしも後処理オペレーションの後、フレーム化された文字が依然として拒否されるなら、文字のイメージが強化され、２進型式に変換される（これについては後述する）。次に上記の認識プロセスが再び行われる。もしも文字が同定される（上記前提条件を満足する）と、システムは次の文字について上記処理（位置特定およびフレーム化）を行う。

【００３６】中央位置のみを検査すると、前記二つの最近接文字以外の文字であって中央位置外の位置にある文字が近接カウントを有する正しい選択文字である、という可能性を見逃すかも知れない。後処理はそのような状況を修復することができる。もしも選択された二つの最低カウントの文字が共にそれらの各文字についての予定拒否しきい値を超えており、あるいは文字間距離（最低ミスマッチと第二最低ミスマッチとの差）が小さすぎるときは、一層良好な照合結果を得るため多数の他のテンプレートが他の８位置において検査される。いずれのテンプレートを処理するかの選択は、例えばテンプレートどうしの最近接文字の類似性に基づいて決定される。例えば最近接する二つの選択文字（最低の二つのミスマッチカウントを有するもの）が文字ＤとＯであるとき、これらは両方ともそれらの拒否しきい値以上のミスマッチカウントを有する。後処理オペレーションではＤおよびＯに類似する予定の組の文字が次に処理される。これには文字０（ゼロ）、Ｑ、Ｃ及びＵが含まれることがありうる。このオペレーションの後で最近接の二つの文字のミスマッチカウントが検査され（最低のミスマッチカウントが最初に検査される）、それらが拒否条件および文字間距離条件に合うか否かが観察される。もしも少なくとも一文字が条件に合うと、その文字が結果として返され、イメージ中の次の文字の処理（位置特定およびフレーム化）が進められる。

【００３７】もしも上記文字のいずれも拒否条件に適合しないか、あるいは文字間距離が小さすぎると、フレーム化された文字は拒否される。この場合、（後述の通り）そのイメージが強化される、その文字は再度２進化され、第二回目の上記認識プロセスが反復される。もしもその文字が認識されると、その文字が返される。そうでないときは結果として拒否文字コードが返され、文書上の次の文字の処理（位置定めおよびフレーム化）が続けられる。

【００３８】上記の拒否しきい値に加えて後述する出口

しきい値（exit threshold）（例えば値１０）も文字同定プロセスに使用することができる。処理速度を高めるため、後続テンプレートミスマッチカウントが出口しきい値に比較され、出口しきい値未満となるテンプレートミスマッチカウントを持つ最初のテンプレートが当該文字を同定する。この場合、残りのテンプレートは処理されない（これによって処理速度が高まる）。

【００３９】さらに文字同定プロセスの速度を高めるため、テンプレートの各行が処理されているときにもしも現在のテンプレート（すなわち現在処理中のテンプレート）のミスマッチカウントがそれまでに得られた下から二番目の最低カウントよりも大きいことが発見されると、進行中の処理は中止される。なぜならばこの現在ののテンプレートはもはや最低ミスマッチカウントを持つ二つのうちの一つでありえないからである。認識速度をさらに高めるため、テンプレートはそれらが最も起生する確率の高い順序（例えば英数字フォントでは子音を従えた母音の前に数字が来る）に予め順序づけておくこともできる。もしもミスマッチカウントが出口しきい値より低いなら、（上述したように）残りのテンプレートは調べる必要がない。このように、最も頻繁に起こる文字が最初に調べられるべきである。またより高速の文字同定を促進するためには、テンプレートにより中央位置のみ検査し、その後でもしも最低テンプレートミスマッチカウントが出口しきい値を超えれば、出口しきい値に最近接する二つのテンプレート各々について残りの８位置を処理するようにでき、最低のテンプレートミスマッチカウントを持つテンプレートをもって当該文字を表すものとすることができる。さらに同定を速くするため、残りの８位置（図７）は次の順序　７９、８１、８３、８９、９１、９３、９５で処理することができる。これは文字が最も配置されている可能性の高い順序である。

【００４０】システムはフレーム化した文字を同定した後、前にフレーム化した文字の左へ、選択したピクセル数（例えば３）だけセグメント化ウィンドウを移動することにより、次の文字の処理に進む（すなわち当該フレーム化した文字に近接する文書上の領域を表す次のグループの格納済みピクセル値グループを処理する。）次にこの近接領域における文字がフレーム化（セグメント化）され、上述したように同定される。エンド-オブ-フィールドが検出されるまで（すなわちもはや文字がなく、エンド-オブ-フィールド文字があるまで）、残りの文字（すなわち残りの格納済みピクセルデータ）がすべてこのように処理される。

【００４１】もしも文字同定プロセスにおいて文字が拒否されると、同定を容易にするため、そのイメージが強化される。このイメージの強化は例えば一定のストローク幅をもつ文字（すなわち文字の各鉛直ストローク幅または水平ストローク幅が一定数の暗ピクセル数を有する文字）のイメージを発生することにより行うことができる。これは以下に述べるようにして達成できる。第一に、文字を表すグレイスケールピクセル値（例えば６-ビット、８-ビット、または１６-ビット値）はもっと小さな範囲の値に変換される。（この変換は高速処理のため例えば３-ビットグレイスケールピクセル値に変換するものである。）　次いで当該文字イメージの個別ピクセル値が検査され、ストロークのピクセル値のうち最も暗い二つまたは三つの値のみが黒に変えられ、他は白に変えられる。このことにより、例えば米国特許第４,６２５,３３０号に説明されているような一定ストローク幅の２進イメージが発生する。

【００４２】文字同定を改善する一助として、各文字の縁を以下のように処理することができる。上述したように、文字同定プロセスで使用した各テンプレートは選択された文字セット（例えばＥ１３Ｂ文字セット）の一文字に対応し、図２および図４に示すように、各文字は幅１４ピクセル、高さ２１ピクセルである。文字の左および右に余分の一列を付加し、文字の頂部および底部に余分の一行を付加する。これによって文字の大きさは１６×２３となる。各テンプレートは各文字に対応して幅１６ピクセル、高さ２３ピクセルである。各テンプレートは１４列文字の両側（右側および左側）に余分の列を有し、また２１行文字の頂部および底部に余分の一行を有する。各余分の行は１６個の白ピクセルを含み、各余分の列は、２３個の白ピクセルを含む。これら余分の行および列はある種の文字の縁部分の同定を一層よくするための白ピクセル境界で、１４×２１文字領域を囲む役割をする。これについて以下に述べる。例えばもしも文字「Ｅ」が１２×２１「Ｆ」テンプレートと比較照合され、このテンプレートが上方に一ピクセル位置だけ移動されると、「Ｅ」の底部におけるピクセル値は失われる。なぜならばこの底部はテンプレートミスマッチカウントの発生に使用できなくなるからである。これはＥをＦと誤って同定する結果を来たしうる。なぜならばミスマッチカウントが低いからである。１６×２３テンプレートでは、Ｅの底部は失われないので、Ｆテンプレートの底部の白行に比較されるミスマッチカウントに実質的に寄与することにより、ＥをＦと誤同定することを回避できよう。同様の説明は１６×２３テンプレートの右および左列に関しても成り立ち、これらの列は「Ｂ」と「３」のような文字間の誤同定を低減する働きをする。

【００４３】図８に示すように、各１６×２３テンプレートはＲＡＭ２７の６９個の１６ビットワードによって表すことができる（一インチ当たり２００ピクセルの走査に相応する）。６９ワードのうち、２３ワードは第一レイヤービットパターンワードを表し、２３ワードは第二レイヤービット有意ワードを表し、２３ワードが第三レイヤービット重みワードを表す。２３ワードの組の各々が文字の高さに対応し、各１６ビットワードが文字の幅に対応する。（簡単のため、図６では論理的オペレー

ションを表すのに１６ビットではなく８ビットを使用した。）

【００４４】以下に掲げるのは本システムが上記論理的＊オペレーションその他の処理オペレーションを行うことができるための格納プログラムの疑似コードリストである。

疑似コード

```
PROCEDURE:    MAIN
                  文書イメージからコードラインを認識する。
                    while  (NOT_END_OF_DOCUMENT)
                      {
                        if  (FIND_FIRST_CHARACTER finds a character)
                        then
                                RECOGNIZE_FIELD
                        else
                                END_OF_DOCUMENT
                      }
PROCEDURE:    FIND FIRST CHARACTER
                do
                    {
                     SCAN_IMAGE_FOR_WHITE_TO_BLACK_TRANSITION
                    }
                until (FOUND_CHARACTER  or  END_OF_DOCUMENT)
PROCEDURE:    RECOGNIZE FIELD
                      do
                          {
                            RECOGNIZE_CHARACTER
                            move_to_next_position
                          }
                        while  (FOUND_CHARACTER)
PROCEDURE:    RECOGNIZE CHARACTER
                  BINARIZE_CHARACTER_IMAGE
                  TEMPLATE_RECOGNITION_OF_CHARACTER
                    if REJECTED      then
                      {
                       ENHANCED_BINARIZE_CHARACTER_IMAGE
                       TEMPLATE_RECOGNITION_OF_CHARACTER
                      }
```

【００４５】バーコード文字データの処理

非バーコード文字の認識に関する上記のテンプレートマッチングオペレーションは、いくつかの修正（下記）を施すことにより、バーコード文字データに適用可能である。

【００４６】前述したように文書上のバーコード文字を走査し、グレイスケールピクセル値を発生させ、メモリ２７に格納した後で、格納済みのピクセル値が文字毎に他の文字のピクセル値からセグメント化される。文字はセグメント化され、文字の底部右縁を表すＸ-Ｙ座標を発生する。次にこの座標は処理を受けるため、セグメント化ルーチンからバーコード文字認識ルーチンへ送信される（渡される）。もしもこのバーコード認識ルーチンが当該バーコード文字を適切に認識することに失敗すると、次にこのＸ-Ｙ座標文字データを処理するための、テンプレートマッチング文字認識ルーチン（template matching character recognition routine）が呼び出される。（この代わりとしてＸ-Ｙ座標データを直接にテンプレートマッチング文字認識ルーチンに渡し、バーコード認識ルーチンを素通りしてもよい。）　テンプレートマッチングルーチンが呼び出されると、当該文字の底部右隅位置のＸ-Ｙ座標がこのルーチンに渡される。（Ｘ座標は文字の水平座標すなわち列座標を表し、Ｙ座標は鉛直座標すなわち行座標を表す。）　次にテンプレートマッチングルーチンは以下に述べるようにこのＸ-Ｙ座標を使ってバーコードセグメンテーションウインドウ（bar-code segmentation window）を設定する。このセグメンテーションウインドウは水平位置（座標）に関して ＣＭＣ７文字幅±１文字幅、鉛直位置（座標）に関して文字高±１／２文字高の範囲内に文字位置を特定

するように拘束される。（これに失敗すると、バーコードセグメントルーチンにより決定されたX-Y座標位置は、分類化に使用される。）　分類化に先立ち、以下に説明するように文字の境界が検査される。（これはテンプレートマッチングルーチンを呼び出す前に行われる。）

【００４７】ＣＭＣ７文字の検査は、当該文字を囲んでいる１ピクセル幅の境界の中のピクセル値の平均値を決定することにより、当該文字が明確な境界を有するか否かを決定するために行われる。もしもこの平均グレイスケールピクセル値が以下に定義するしきい値（ｔｈｒ）を超えると、それは文字が著しく崩れていることを示し、テンプレートマッチングは行われない。そうでないときはテンプレートマッチングルーチンが呼び出され、

ｔｈｒ＝０．９＊（（黒＋白）／２）

で定義されるしきい値を使ってテンプレートマッチングが行われる。ここに「黒」および「白」は当該文字を予備走査（prescanning）することにより決定される。

「黒」は当該文字の鉛直バー上の最高レベルピクセルであり、「白」は鉛直バー間に位置する最低レベルピクセルである。バーコード文字データは、バーコード文字の縁を決定するために当該文字の内側および外側の二つの連続列が検査されることを除けば、（上述した）非バーコード文字データの場合と実質上同様にセグメント化（フレーム化）される。バーコード文字の性質のため、（非バーコード文字について行われるように）ウインドウの内側および外側の単一列を使用するとその単一列がバー間の白い間隙に落ちる可能性があり、そうなると当該文字のセグメントメンテーション（segmentation、セグメント化)を誤まる。そのような過誤セグメンテーションを回避するため、図４（ｂ）に示すようにウインドウ１５３の左の三列１７２内のピクセルの和（和１７２）がウインドウ内の右の三列１７１内のピクセル和（和１７１）と比較される。もしも和１７２＞和１７１であると、ウインドウを左へ移動することにより正味の利得が現われる。もしもウインドウの右の三列１７３内のピクセル和がウインドウ内の左の三列１７０内のピクセル和を超えると、ウインドウを右へ移動することにより正味の利得が現われる。上記いずれの状況も真である場合は以下の計算が行われる。

もしも　（和１７２－和１７１）＞（和１７３－和１７２）　なら、ウインドウは左へ移動される。

もしも　（和１７２－和１７１）＜（和１７３－和１７２）　なら、ウインドウは右へ移動される。

このようにしてグレイスケールピクセル値の和（Ｓｘ）は最大値をとるに至る。バーは鉛直方向に向いているので、文字の上方および下方の単一行和はウインドウの鉛直方向の移動を決定するのに適切であり、これは非バーコード文字の場合と同じである。文字データがメモリ２７に適切にセグメント化されているとき（すなわち文字がウインドウ１５３内で適切にフレーム化されているとき）は、グレイスケールピクセル値すべての和は最大値（Ｓｗ）をとる。

【００４８】セグメント化された各文字は走査データ値からなるアレイ（配列）（幅２１ピクセル、高さ２８ピクセル）として表される。鉛直走査線３７（図１）に沿うデータはグレイスケールピクセル値の波形を表す。ウインドウ内のイメージを２進化するための適切なしきい値を発見するため、イメージを検査して最も黒い（２５５に最も近い）ピクセルと最も白い（０に最も近い）ピクセルとが見い出される。次に以下の方程式に基づいて２進化しきい値が設定される。

しきい値＝最も黒いピクセル－（最も黒いピクセル－最も白いピクセル）＊２／３

例えばもしも最も白いピクセルが値６０を有し、最も黒いピクセルが値２００を有するなら、しきい値は

しきい値＝２００－（２００－６０）＊／３＝１０７

のレベルに設定されることになろう。このしきい値レベルはしきい値を超えるセグメント化済み文字データのすべてのグレイスケールピクセル値を「１」（黒を表す）に設定し、しきい値未満のグレイスケール値をすべて「０」（白を表す）に設定することにより、セグメント化済み文字データを２進化するのに使用される。これによって当該セグメント化された当該文字を表す２進（０,１）ピクセル値からなる２１列（２１ピクセル幅）×２８行（２８ピクセル高）アレイを生ずる。

【００４９】「非バーコード文字データの処理」と見出しを付けた項に上述したテンプレートマッチングルーチン手順を使用して文字の認識を行う準備として、テンプレートの大きさに相当するように上記２進アレイの大きさが低下される（圧縮される）。例えばアレイの底部行から開始して、ピクセル行を隔行に削除または省略する（アクセスを跳ばす）ことにより、アレイは１４行×２１列のピクセルアレイに低減（圧縮）できる。さらに圧縮したアレイを反時計方向に９０度回転することにより（すなわちアレイが回転されたかのごとくにアレイのデータ値にアクセスすることにより）、アレイは２１×１４アレイと見ることができ、実際、各文字の境界を形成する白ピクセル（文字の両側それぞれと上側および下側に一ピクセル）が含まれるときは、２３×１６アレイと見ることができる。このアレイの圧縮および回転によりバーコード情報（バーと間隙の比）を著しく損なうことなくアレイの大きさおよび方向をテンプレートの大きさおよび方向に相当するようにすることができる。

【００５０】次に、メモリ２７内に格納された、予定バーコード文字パターンを表す相当サイズ（２３×１６）のテンプレートを上述したように使用して当該文字の２進パターンをテンプレートパターンに符号させ、当該文字を認識する。図９（ａ）には欠陥あるバーコード情報を持つ文字（文字「４」）の例が示してある。それに対

応する、（欠陥を表すノイズを含む）２進パターンが図９（ｂ）に示されている。文字「４」の認識に使用される符合するテンプレートのビットパターンの例が図９（ｃ）に示されている。もしもテンプレートマッチングルーチンが文字の認識に失敗すると、当該文字の２進化に使用したしきい値が

しきい値＝最も黒いピクセル－（最も黒いピクセル－最も白いピクセル）＊１／２

に低下される。前述の例と同様に、もしも最も白いピクセルが値６０を持ち、最も黒いピクセルが値２００を持つことが発見されたとすると、しきい値は

２００－（２００－６０）＊１／２＝１３０

のレベルにされる。当該文字情報がＣＭＣ７文字のバーよりも黒でない（すなわち低いピクセル値を有する）ときはこのしきい値レベルを使用することによりＣＭＣ７文字を不明瞭化する情報（例えばスタンプ、汚れ、べん書きマーク等）を「脱落させる」効果を有する。

【００５１】

【効果】本テンプレートマッチングシステムは文字全体の形状またはパターンを検査するので、バーに欠損がある場合、余分なバー（署名によるペン書きマーク等）を含む場合、あるいは文字パターンのバー間隔に不規則その他の文字崩れの問題がある場合でも文字認識ができる。上述したように、本発明のテンプレートマッチングシステムをバー／間隙比分析システムと関連させて（またはその代わりに）使用するとき、読み取り速度（認識すべき全文字数に対する認識された文字数の比率）が著しく改善される。

【図面の簡単な説明】

【図１】本発明のシステムのブロック線図である。

【図２】図１のシステムにより読み取り可能（走査可能かつ同定可能）なバーコード文字の外観を例示する図で＊ある。

【図３】いろいろの不透明さ（グレイスケール値）をもつピクセルからなるマトリックスを有する、走査した文字の図解である。

【図４】バーコード化されていない文字のフレーム化に使用されるウィンドウ（ａ）およびバーコード化されている文字のフレーム化に使用されるウィンドウ（ｂ）の図である。

【図５】バーコード化されていない文字のフレーム化に使用した図４（ａ）のウィンドウの図である。

【図６】走査およびフレーム化した文字を表す２進データにいろいろの論理オペレーションを行った結果を示すブロック線図である。

【図７】フレーム化した文字のイメージのいろいろの部分を表すマトリックスのブロック線図である。

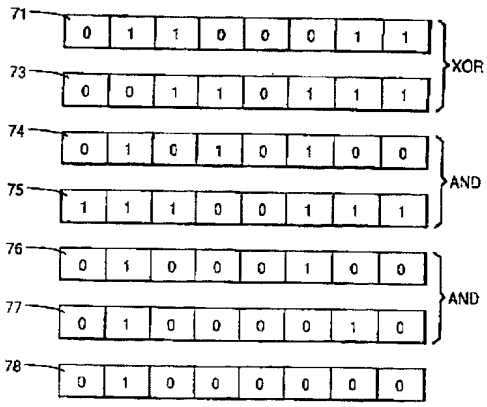
【図８】本システムがバーコード化されていない文字を表すのに使用するテンプレートおよび関連の２進値の図解および表である。

【図９】印刷品質の低いバーコード文字、対応の２進パターン、およびバーコード文字を表すのに図１のシステムが使用した対応のテンプレートを例示する図である。

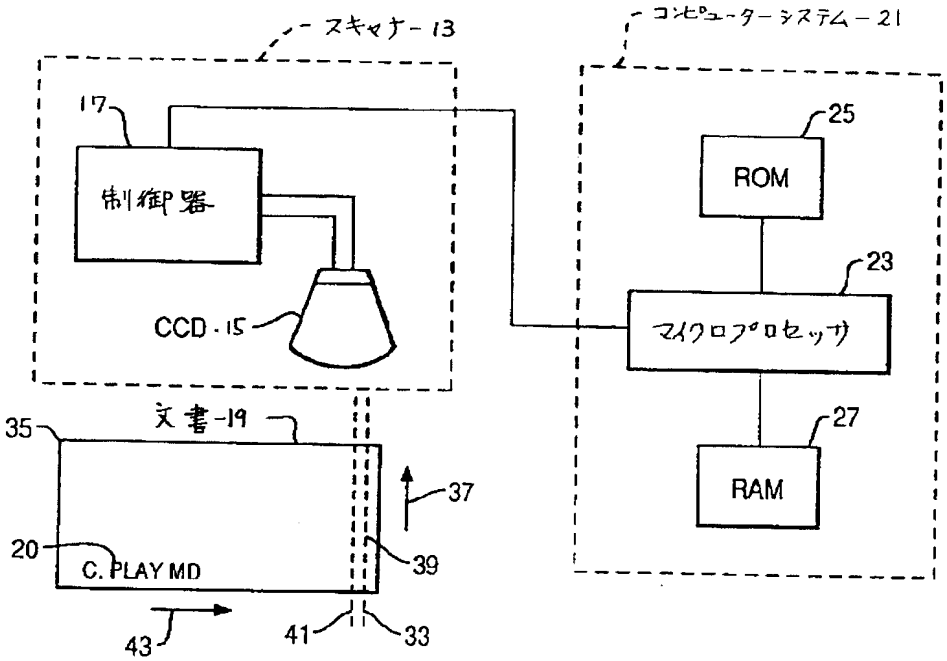
【符号の説明】

２０　英数字	３３　文書の右底部隅
３５　文書の頂部左隅	３７　走査方向
３９　最右の走査線	４１　走査線
４３　文書の移動方向	４５　鉛直バー（データバー）
４７　短い間隙	４９　長い間隙
５１　暗ピクセル	５３　明ピクセル
５５　文字間空間	

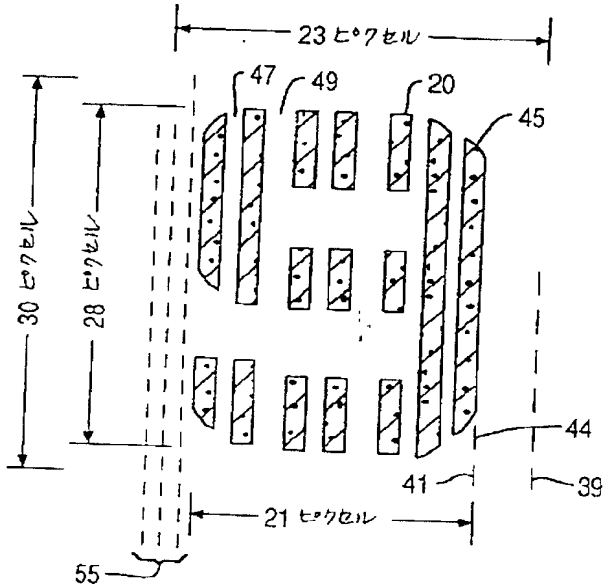
【図６】



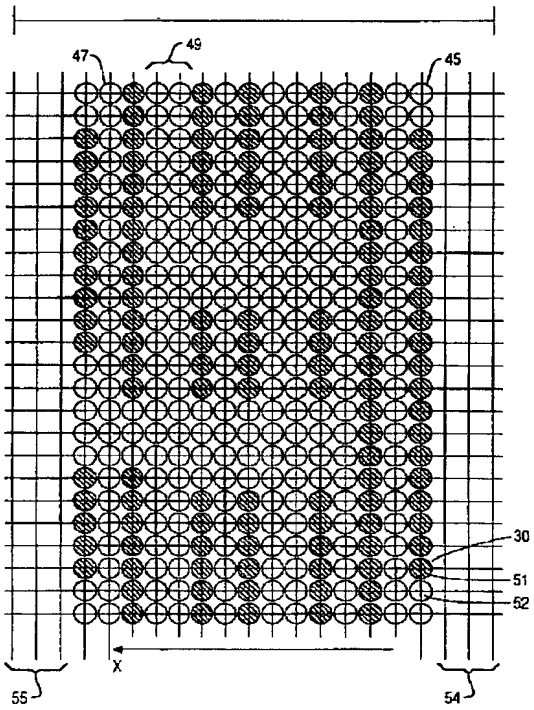
【図1】


スキャナー13
コンピューターシステム－21
17
制御器
CCD-15
25
ROM
23
マイクロプロセッサ
27
RAM
35
文書-19
37
20
C. PLAY MD
39
43
41
33

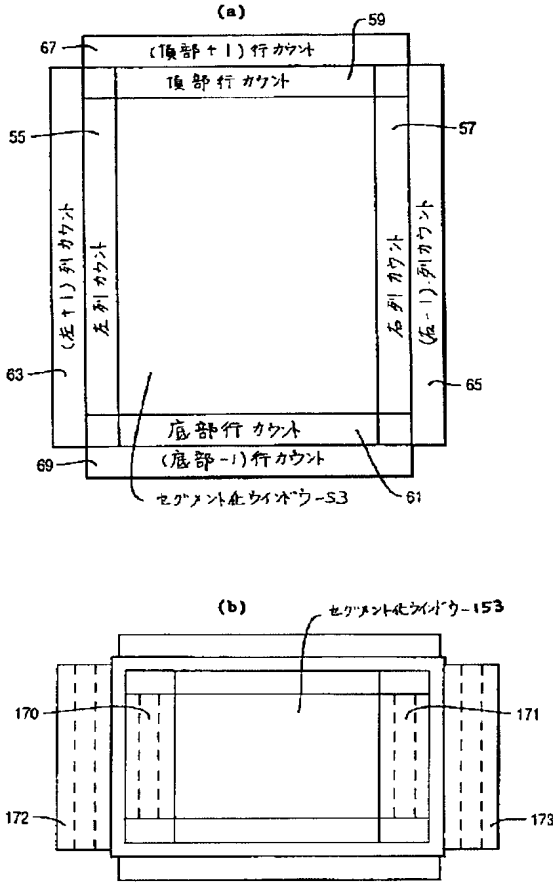
【図2】


23ピクセル
47
49
20
45
30ピクセル
28ピクセル
44
41
39
21ピクセル
55

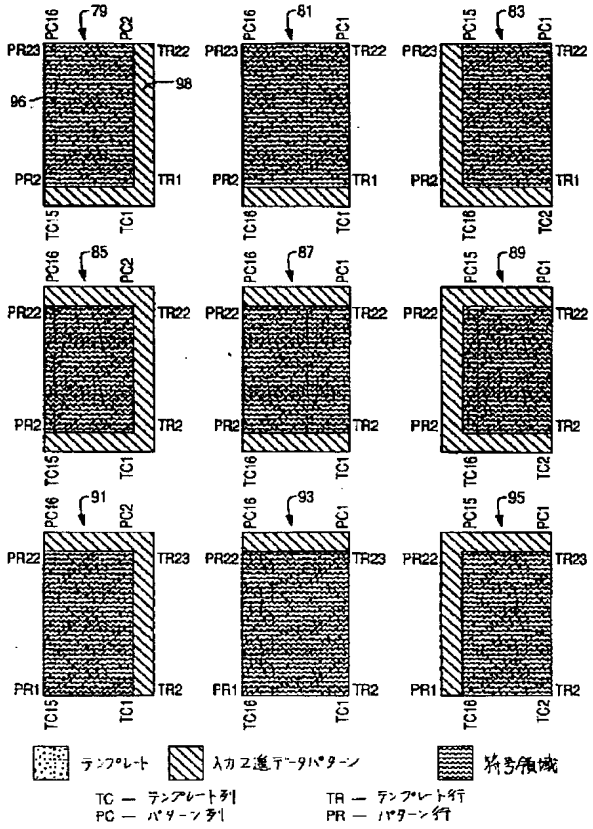
【図3】


47
49
45
30
51
52
X
55
54

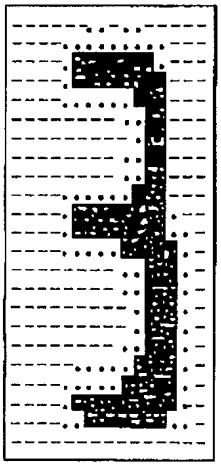
【図4】



【図7】

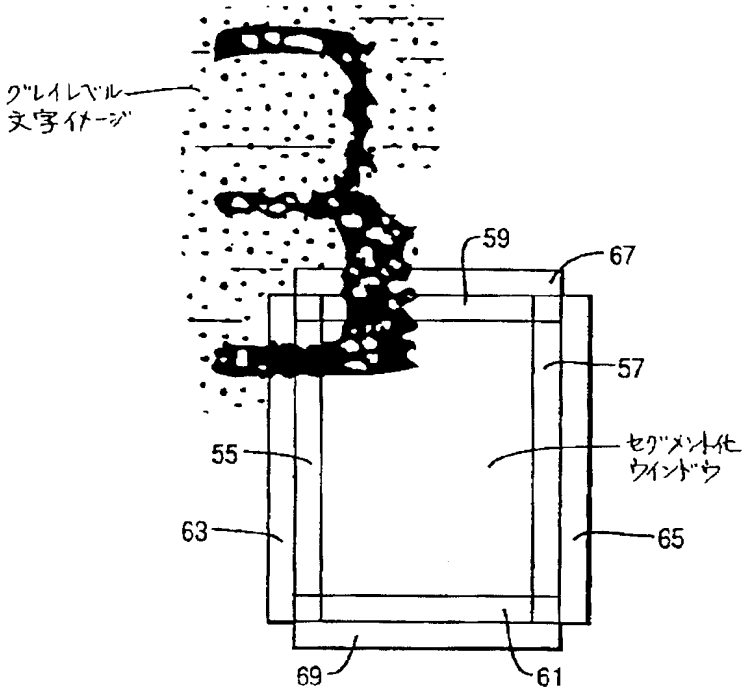


【図8】

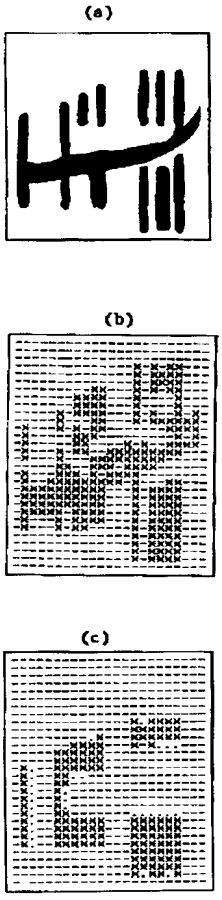


パターンワード (16進法)		ビット有意ワード (16進法)		重みワード (16進法)	
00	00	F0	9F	00	00
00	00	F0	07	00	00
07	F0	F7	F7	00	00
07	F6	F7	FF	00	00
00	36	F0	3F	00	00
00	18	FF	9F	00	00
00	18	FF	DF	00	00
00	18	FF	9F	00	00
00	18	FF	9F	00	00
00	36	F0	3F	00	00
07	F8	F7	F8	60	E0
07	F8	F7	F9	60	E0
00	7C	F0	7D	00	00
00	1C	FF	9D	00	00
00	1C	FF	9D	00	00
00	1C	FF	DD	00	00
00	1C	FF	DD	00	00
00	1C	FF	9D	00	00
00	3C	F8	3D	00	00
00	7C	F0	7D	00	00
07	FC	F7	F0	00	00
03	F8	F3	F8	00	00
00	00	FF	FF	00	00

【図５】



【図９】



フロントページの続き

(72)発明者　メアラ　クルカーニ
カナダ　エヌ２ティー　１ピー３、オンタリオ、ウォータールー、リージェンシー　クレッセント　463

(72)発明者　レイモンド　エル．ヒギンズ
カナダ　エヌ２エヌ　２エム１、オンタリオ、キッチナー、イエロー　バーチ　ドライブ　25